WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Saturday, March 20, 2004

Hide?	Set Name	Query	Hit Count
	DB=EPAB,DWPI; PLUR=YES; OP=ADJ		
☐	L14	L13 and ((server near4 request$) near8 (distribut$ or balanc$))	3
☐	L13	(server near2 (capabilit$ or capacit$ or characteri$)) same (server near4 (select$ or assign$))	31
	DB=USPT; PLUR=YES; OP=ADJ		
☐	L12	L11 and ((server near4 request$) near8 (distribut$ or balanc$))	43
☐	L11	(server near2 (capabilit$ or capacit$ or characteri$)) same (server near4 (select$ or assign$))	182
☐	L10	6205477[uref]	4
☐	L9	l7 and l8	7
☐	L8	(709/226 or 709/203 or 709/220 or 709/223 or 718/105 or 718/104 or 718/102).ccls.	5428
☐	L7	l5 same l1	8
☐	L6	L5 and l1	43
☐	L5	l3 same L4	182
☐	L4	server near4 (select$ or assign$)	7024
☐	L3	server near2 (capabilit$ or capacit$ or characteri$)	2240
☐	L2	6205477[pn]	1
☐	L1	(server near4 request$) near8 (distribut$ or balanc$)	753

END OF SEARCH HISTORY

US006601084B1

(12) **United States Patent**
Bhaskaran et al.

(10) **Patent No.: US 6,601,084 B1**
(45) **Date of Patent: Jul. 29, 2003**

(54) **DYNAMIC LOAD BALANCER FOR MULTIPLE NETWORK SERVERS**

(75) Inventors: **Sajit Bhaskaran**, Sunnyvale, CA (US); **Abraham R. Matthews**, San Jose, CA (US)

(73) Assignee: **Avaya Technology Corp.**, Basking Ridge, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/992,038**

(22) Filed: **Dec. 19, 1997**

(51) **Int. Cl.**^{7} **G06F 9/00**
(52) **U.S. Cl.** **709/105**; 709/223
(58) **Field of Search** 709/105, 223–226, 709/229

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,283,897 A	* 2/1994	Georgiadis et al.	709/105
5,473,599 A	12/1995	Li et al.	370/219
5,586,121 A	12/1996	Moura et al.	370/404
5,612,865 A	* 3/1997	Dasgupta	700/79
5,612,897 A	3/1997	Rege	709/219
5,634,125 A	* 5/1997	Li	707/203
5,687,369 A	* 11/1997	Li	707/203
5,754,752 A	5/1998	Sheh et al.	714/4
5,796,941 A	* 8/1998	Lita	713/201
5,812,819 A	9/1998	Rodwin et al.	703/23
5,815,668 A	9/1998	Hashimoto	709/238
5,835,696 A	11/1998	Hess	714/10
5,936,936 A	8/1999	Alexander, Jr. et al.	370/216
5,951,634 A	* 9/1999	Sitbon et al.	709/105

FOREIGN PATENT DOCUMENTS

WO	WO 99/32956	7/1999	G06F/0/00

OTHER PUBLICATIONS

Internet. "Quasi–Dynamic Load–Balancing (QDLB) Methods." Apr. 25, 1995, pp. 2 and 5.
(INTERNET) "Dynamic Load–Balancing Methods", Apr. 25, 1995.*
(Becker) Becker, Wolfgang. "Dynamic Load Balancing for Parallel Database Processing", 1997.*
(Omiecinski) Omiencinski, Edward. "Performance Analysis of a Load Balancing Hash–Join Algorithm for a Shared Memory Multiprocessor", 1991.*
(IBM) "Value–Oriented Approach fto Selecting Buckets for Data Redistribution", May 1, 1993.*

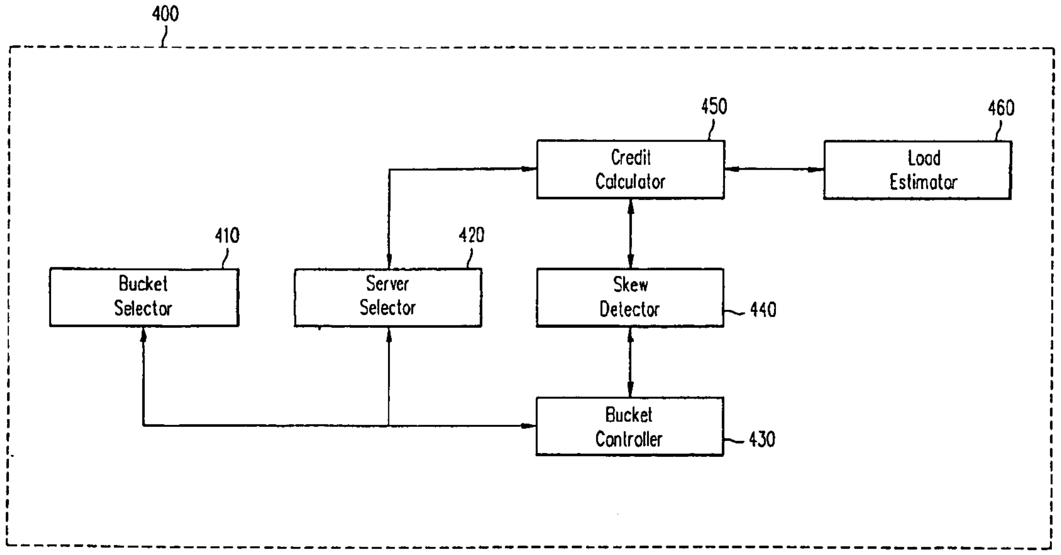
* cited by examiner

Primary Examiner—Alvin Oberley
Assistant Examiner—Lewis A. Bullock, Jr.
(74) *Attorney, Agent, or Firm*—David Volejnicek

(57) **ABSTRACT**

The present invention provides methods and systems for balancing the load on a plurality of servers using a load balancing algorithm which continuously examines the loads on the plurality of servers and makes adjustments in the loads accordingly. Among the factors considered in the load balancing are the power of each server relative to other servers, the load on each server relative to the other servers, and a "credit" for each server based on their power and load.

37 Claims, 10 Drawing Sheets



First Hit Fwd Refs

☐ Generate Collection Print

L19: Entry 2 of 13 File: USPT Jul 29, 2003

DOCUMENT-IDENTIFIER: US 6601084 B1
TITLE: Dynamic load balancer for multiple network servers

Detailed Description Text (36):
At time epoch t, load estimator 460 gets the load vector and converts it into a scalar quantity L.sub.i. Since the servers are of different computing capacity, load estimator 460 normalizes the loads before they can be compared. Intuitively, a load of 100 units to a server of capacity 10 should equal a load of 200 units to a server of capacity 20. Thus, load estimator 460 normalizes based on the inverse of the server weights. Server credits are computed periodically to decide which server has the maximum capacity available. In Table 1, rows 3 to 7, show the computation of server credits by credit calculator 450 at epoch t when the raw loads are all equal. Row 3: Current load per server (L.sub.i). This is computed from a load vector. Row 4: The weighted load for each server is computed in accordance with equation 2. Note that even though each server has the same load, it represents a larger load for server 660 than for server 670 or server 680. The effective load for an idealized server is the sum of the normalized loads for all servers; Row 5: To relate the load to capacity, the load normalizer LN is computed in accordance with equation 3. Evidently, this value has to be a function of the current load. Row 6: A normalized estimate of computational resources used by the current load is subtracted from the server's weight to arrive at the server's credits (K.sub.i) in accordance with equation 4. Note that the idealized server's credits can be computed in two different ways. In one method, we use the same formula that we use for each server. Alternately, we could sum up the credits of all servers. Both methods should come out to be equal (or very nearly equal, due to integer arithmetic). Row 7: Finally, an estimate of computational use by a new flow is needed for each server. Towards this goal, we compute the flow weight in accordance with equation 5. This is also the amount that we must deduct from the idealized server's credits when a flow is assigned to it. Whenever a flow is assigned to a server at a later time epoch (say t+k), the computational resource to be subtracted from the server's credits is shown in Table 1, rows 8 (in accordance with equation 6). This credit adjustment is performed by server selector 420 whenever a bucket-to-server assignment occurs. Row 8: The computational resource that a flow consumes for each server is computed in accordance with equation 6. This is the value to deduct from the server's credits when a new bucket is assigned to the server. Note that we can assign three flows to server 680, two flows to server 670, and, one flow to server 660, and each individual server's credits would be diminished by the same amount.

Detailed Description Text (39):
As explained above, load estimator 460 provides credit calculator 450 with an estimate of the load on each server. Common measures of the load on a server are based on the number of data requests, the number of data packets received by the server, the number of bytes received by the server, the number of data packets sent by the server, the number of bytes sent by the server, the processor activity on the server, and the memory utilization on the server. Because load balancer 400 is primarily designed to balance the network load on a plurality of servers, load estimator 460 typically uses load measures based on network traffic to the server. Thus, most embodiments of load estimator 460 use load measures based on the number of data requests received by the server, the number of data packets received by the

server, the number of bytes received by the server, the number of data packets sent by the server, or the number of bytes sent by the server.

CLAIMS:

28. The method of claim 20 wherein: a load of a server comprises a plurality of ones of: a number of data requests received by the server, a number of data packets received by the server, a number of bytes received by the server, a number of data packets sent by the server, and a number of bytes sent by the server.

37. The load balancer of claim 29 wherein: a load of a server comprises a plurality of ones of: a number of data requests received by the server, a number of data packets received by the server, a number of bytes received by the server, a number of data packets sent by the server, and a number of bytes sent by the server.

US005603029A

United States Patent [19]
Aman et al.

[11] Patent Number: 5,603,029
[45] Date of Patent: Feb. 11, 1997

[54] SYSTEM OF ASSIGNING WORK REQUESTS BASED ON CLASSIFYING INTO AN ELIGIBLE CLASS WHERE THE CRITERIA IS GOAL ORIENTED AND CAPACITY INFORMATION IS AVAILABLE

[75] Inventors: **Jeffrey D. Aman**, Wappingers Falls, N.Y.; **Curt L. Cotner**, Gilroy, Calif.; **Donna N. T. Dillenberger**, Yorktown Heights; **David B. Emmes**, Poughkeepsie, both of N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **476,157**

[22] Filed: **Jun. 7, 1995**

[51] **Int. Cl.**^{6} **G06F 15/06**

[52] **U.S. Cl.** **395/675**; 395/616; 395/859; 395/200.01; 364/242.94; 364/229; 364/230.3; 364/286; 364/281; 364/284.4

[58] **Field of Search** 395/200.01, 200.03, 395/600, 650, 826; 364/242.94, 229, 230.3, 280, 281, 284.4, 902

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,481,583	11/1984	Mueller	395/732
5,241,677	8/1993	Naganuma et al.	395/650
5,341,477	8/1994	Pitkin et al.	395/200
5,537,542	7/1996	Eilert et al.	395/184.01

OTHER PUBLICATIONS

D. Sitaram et al, Issues in the . . . Load Skew, Parallel and Distributed Information Systems, pp. 214–223 1993.

Lin et al., "A Dynamic Load Balancing Policy with a Central Job Dispatcher (LBC)", *Proceedings of the Eleventh International Conference on Distributed Computing Systems, Arlington, Texas,* 20–24 May 1991, pp. 264–271.

Geise, "Dynamic Load Balancing in a Cluster of Loosely Coupled Systems", *IBM Technical Disclosure Bulletin,* vol. 37, No. 7, Jul. 1994, pp. 243–244.

Herrin II, et al., "The Benefits of Service Rebalancing", *Proceedings of the Third Workship on Workstation Operating Systems,* Key Biscayne, FL, 23–24 Apr. 1992, pp. 104–110.

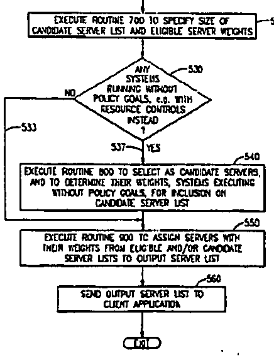
Menditto et al., "Single System Image And Load Balancing For Network Access To A Loosely Coupled Complex", *IBM Technical Disclosure Bulletin,* vol. 34, No. 9, Feb. 1992, pp. 464–271.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Anderson I. Chen
Attorney, Agent, or Firm—William A. Kinnaman, Jr.; Peter L. Michaelson; Michaelson & Wallace

[57] **ABSTRACT**

Apparatus and accompanying methods for use preferably in a multi-system shared data (sysplex) environment (**100**), wherein each system (**110**) provides one or more servers (**115**), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system(s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

44 Claims, 25 Drawing Sheets


ENTER
(ROUTING SELECTION REQUEST FROM CLIENT APPLICATION)
500
EXECUTE ROUTINE 600 TO DETERMINE ELIGIBLE AND CANDIDATE SYSTEMS & SERVERS
510
EXECUTE ROUTINE 700 TO SPECIFY SIZE OF CANDIDATE SERVER LIST AND ELIGIBLE SERVER WEIGHTS
520
ANY SYSTEMS RUNNING WITHOUT POLICY GOALS, e.g. WITH RESOURCE CONTROLS INSTEAD ?
530
NO
533
537
YES
EXECUTE ROUTINE 800 TO SELECT AS CANDIDATE SERVERS, AND TO DETERMINE THEIR WEIGHTS, SYSTEMS EXECUTING WITHOUT POLICY GOALS, FOR INCLUSION ON CANDIDATE SERVER LIST
540
EXECUTE ROUTINE 900 TO ASSIGN SERVERS WITH THEIR WEIGHTS FROM ELIGIBLE AND/OR CANDIDATE SERVER LISTS TO OUTPUT SERVER LIST
550
SEND OUTPUT SERVER LIST TO CLIENT APPLICATION
560
EXIT

First Hit Fwd Refs

☐ Generate Collection Print

L19: Entry 12 of 13 File: USPT Feb 11, 1997

DOCUMENT-IDENTIFIER: US 5603029 A
TITLE: System of assigning work requests based on classifying into an eligible class where the criteria is goal oriented and capacity information is available

Abstract Text (1):
Apparatus and accompanying methods for use preferably in a multi-system shared data (sysplex) environment (100), wherein each system (110) provides one or more servers (115), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system(s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

Brief Summary Text (28):
However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement may be to rotate the server entries among the different systems. Owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Brief Summary Text (29):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight. The client may request a refresh of this balancing assessment on a periodic basis.

Detailed Description Text (24):

However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement here may be to rotate servers around different systems before the same system is selected again for a different server. In any event, owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Detailed Description Text (25):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight.

Detailed Description Text (35):
Once routine 600 completes processing, what remains is two lists of servers: a list of eligible servers, i.e., policy related servers for which capacity information is available, and a list of all candidate servers--the former list possessing available requisite capacity at the lowest business importance level, the latter list being servers, whether goal-oriented or not (i.e., running under a policy or being "black box") but for which no capacity information is available. Routine 700 is then executed, as noted above, to set a maximum limit on the number of servers that can be selected from either of the server lists and subsequently provided to the client application, and to determine the weights for the eligible servers.

Detailed Description Text (37):
Upon entry into routine 700, execution first proceeds to blocks 705-725 which collectively specify the maximum number of servers that can be selected from the candidate server list and subsequently provided to the client application. Specifically, execution first proceeds to block 705 which, when executed, sets the size of a list of output servers to be provided, by our inventive process, to the client application. This size (variable CANDIDATE.sub.-- COPY.sub.-- SIZE) is taken to be a minimum of the size, as specified by the client application itself in its work request, of its server list, i.e., variable CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE, or the size of the candidate server list itself, i.e., variable CANDIDATE.sub.-- SERVER.sub.-- LIST SIZE. Thereafter, decision block 710 determines whether the eligible server list contains any servers. If not, then the only servers that can be selected are those in the candidate server list. Accordingly, if no eligible servers exist, i.e., ELIGIBLE.sub.-- SERVER.sub.-- LIST is empty, then execution exits, via YES path 712, from routine 700. Alternatively, if eligible servers do exist, then decision block 710 directs execution, via NO path 714, to decision block 715. This latter decision block, when executed, ascertains whether more servers exist in the eligible server list than are requested by the client, i.e., whether CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE.gtoreq.ELIGIBLE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE? If this is true, then only eligible servers and no candidate servers will be placed in the client server list. In this case, decision block 715 will route execution, via YES path 716, to block 720 which, when executed, will set the number of servers, specified in variable CANDIDATE.sub.-- COPY.sub.-- SIZE, to be selected from the candidate server list to zero. Alternatively, if an insufficient number of servers exists in the eligible server list to satisfy the requirements of the client application, thereby requiring some or all of the servers to be selected from the candidate server list, decision block 715 will route execution, via NO path 718, to block

725. This latter block, when executed, will set the number of servers to be selected from the candidate server list, i.e., as specified in variable CANDIDATE.sub.-- COPY.sub.-- SIZE, as the minimum of either the size of the candidate server list itself, i.e., as specified in variable CANDIDATE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE, or the remaining number of servers requested by the client application after all the eligible servers have been selected, i.e., CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE - ELIGIBLE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE. Once block 725 has executed, then execution proceeds to decision block 730.

Detailed Description Text (38):
Blocks 730-775 collectively determine the weights for each of the eligible servers. Decision block 730 determines, based upon the number and type (candidate vis-a-vis eligible) of the presently available servers, whether a relatively large number of candidate, presumably black box servers, would be provided, i.e., servers for which no capacity information is currently known. If this is the case and particularly if less than half of these collectively available systems have eligible servers, then to avoid taking undue risks in assigning work to all these servers, equal weights of one are set for all the servers that will be returned, from both server lists, to the client application. Alternatively, if more than half of these collective systems have eligible servers, i.e., for which capacity information is available, then, with relatively little risk, the weights assigned to the eligible servers can be extended, as discussed below, to the candidate servers that will be returned to the client application. Accordingly, if more than half of the collective systems have candidate servers, then decision block 730 directs execution via its YES path 734, to block 735 which sets a flag, FIXED.sub.-- WEIGHT, which signifies that fixed weights are to be assigned to the candidate servers, to YES. Alternatively, if more than half of the collective systems have eligible servers, then, decision block 730 routes execution, via its NO path 732, to block 740 which, when executed, sets the FIXED.sub.-- WEIGHT flag to NO; hence, variable weights are subsequently set. Once this flag has been duly set by either block 735 or 740, execution proceeds to block 745.

Detailed Description Text (64):
Block 1215, when executed, sets variable SYSTEM to designate the first system in REMAINING.sub.-- SYSTEMS.sub.-- LIST. Thereafter, block 1215 removes the system now designated by variable SYSTEM, i.e., the current system, from REMAINING.sub.-- SYSTEMS.sub.-- LIST. Once this occurs, execution proceeds to decision block 1220 which tests whether the current system is presently active. If this system is not active, then this system is no longer considered with execution merely looping back, via NO path 1224 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if the current system is active, then decision block 1220 routes execution, via its YES path 1222, to decision block 1225. This latter decision block determines, from the GRRI table, whether that system has a real instance that supports the generic resource requested by the client application, i.e., whether an eligible application server resides on the system for the client application. If no such server exists, then the current system is not considered further with execution looping back, via NO path 1228 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. In the event that the current system does possess an eligible application server (real instance for the requested generic resource), then decision block 1225 routes execution, via its YES path 1226, to decision block 1230. This latter decision block ascertains whether this present system is goal-oriented, i.e., running in a goal mode. As noted above, all servers for which capacity information is not available, including non-goal oriented servers, are not selected at all for session placement. In any event, if the current system is not running in goal mode, then decision block 1230 routes execution, via NO path 1233, to block 1234 which, in turn, sets the eligible system list to empty. Execution then exits. Alternatively, if the current system is running in goal mode, then decision block 1230 routes execution, via YES path 1232,

to decision block 1235. This latter decision block ascertains whether the current system, i.e., as identified in SYSTEM, which supports the desired generic resource, has an application server (real instance) that is meeting its goals. If no such application server residing on the current system is meeting its corresponding goals, then decision block 1235 routes execution, via NO path 1238, to decision block 1240. This latter decision block checks to determine whether any of the systems within the set of ELIGIBLE.sub.-- SYSTEMS, as previously chosen by this loop thus far, contains a system that does not have an application server that is meeting its goals. If such a system does not exist, i.e., at least one system has already been chosen that is meeting its goals and the current system is an inferior choice to the system already chosen, then the present system is not considered any further. In this case, execution then loops back, via NO path 1244 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if such a system exists, i.e., the current as well as all the systems selected thus far are not meeting their goals, then decision block 1240 routes execution to block 1245 which, when executed, adds the identification of the current system to the set of selected eligible systems as it presently exists, i.e., ELIGIBLE.sub.-- SYSTEMS.rarw.ELIGIBLE.sub.-- SYSTEMS+SYSTEM. Once the list of eligible systems is updated, then execution then loops back, via path 1246 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on.

Detailed Description Text (86):
Accordingly, in selecting competing servers for session placement, a more refined selection criteria can be used than that described above, with the order of preferential server selection being: (a) first, those servers that are fully meeting their goals; (b) second, those servers that are marginally missing their goals but having requisite capacity at a sufficiently high numeric (low business) importance level, such as level six, or greater, or just unused capacity; (c) third, those servers that are meeting their goals but only have sufficient capacity at a higher (numerically lower) business importance level than those servers marginally missing their goals; and (d) fourth and finally, those servers that are not meeting their goals at all, i.e., by more than a marginal amount. By favoring those servers that are marginally missing their goals (servers (b)) but have capacity at a sufficiently low business importance level to receive new sessions (and work) over those goal-meeting servers that have capacity only at higher business importance levels (servers (c)), this criteria assures that work of higher business importance currently active or pending at the latter servers will not be readily displaced. Those marginal servers that do not possess requisite minimum capacity at a sufficiently low business importance level are classified in the last group of servers. Servers in this last group (servers (d)) are only selected as a last resort since any additional sessions placed thereat (with ensuing work requests) are likely to excessively burden these servers, thus degrading their throughput to the ultimate detriment of meeting the overall business goals of the sysplex. Here, too, servers in each of the four classes (a-d) would be chosen in ascending order of session counts, i.e. those servers having the fewest sessions would be chosen first. The numeric server performance measure (s) that defines "marginal" operation is implementationally dependent and may well be empirically defined.

L19: Entry 12 of 13 File: USPT Feb 11, 1997

DOCUMENT-IDENTIFIER: US 5603029 A
TITLE: System of assigning work requests based on classifying into an eligible class where the criteria is goal oriented and capacity information is available

Abstract Text (1):
Apparatus and accompanying methods for use preferably in a multi-system shared data

(sysplex) environment (100), wherein each system (110) provides one or more servers (115), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system(s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

Brief Summary Text (28):
However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement may be to rotate the server entries among the different systems. Owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Brief Summary Text (29):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight. The client may request a refresh of this balancing assessment on a periodic basis.

Detailed Description Text (24):
However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement here may be to rotate servers around different systems before the same system is selected again for a different server. In any event, owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Detailed Description Text (25):

As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight.

Detailed Description Text (35):
Once routine 600 completes processing, what remains is two lists of servers: a list of eligible servers, i.e., policy related servers for which capacity information is available, and a list of all candidate servers--the former list possessing available requisite capacity at the lowest business importance level, the latter list being servers, whether goal-oriented or not (i.e., running under a policy or being "black box") but for which no capacity information is available. Routine 700 is then executed, as noted above, to set a maximum limit on the number of servers that can be selected from either of the server lists and subsequently provided to the client application, and to determine the weights for the eligible servers.

Detailed Description Text (37):
Upon entry into routine 700, execution first proceeds to blocks 705-725 which collectively specify the maximum number of servers that can be selected from the candidate server list and subsequently provided to the client application. Specifically, execution first proceeds to block 705 which, when executed, sets the size of a list of output servers to be provided, by our inventive process, to the client application. This size (variable CANDIDATE.sub.-- COPY.sub.-- SIZE) is taken to be a minimum of the size, as specified by the client application itself in its work request, of its server list, i.e., variable CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE, or the size of the candidate server list itself, i.e., variable CANDIDATE.sub.-- SERVER.sub.-- LIST SIZE. Thereafter, decision block 710 determines whether the eligible server list contains any servers. If not, then the only servers that can be selected are those in the candidate server list. Accordingly, if no eligible servers exist, i.e., ELIGIBLE.sub.-- SERVER.sub.-- LIST is empty, then execution exits, via YES path 712, from routine 700. Alternatively, if eligible servers do exist, then decision block 710 directs execution, via NO path 714, to decision block 715. This latter decision block, when executed, ascertains whether more servers exist in the eligible server list than are requested by the client, i.e., whether CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE.gtoreq.ELIGIBLE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE? If this is true, then only eligible servers and no candidate servers will be placed in the client server list. In this case, decision block 715 will route execution, via YES path 716, to block 720 which, when executed, will set the number of servers, specified in variable CANDIDATE.sub.-- COPY.sub.-- SIZE, to be selected from the candidate server list to zero. Alternatively, if an insufficient number of servers exists in the eligible server list to satisfy the requirements of the client application, thereby requiring some or all of the servers to be selected from the candidate server list, decision block 715 will route execution, via NO path 718, to block 725. This latter block, when executed, will set the number of servers to be selected from the candidate server list, i.e., as specified in variable CANDIDATE.sub.-- COPY.sub.-- SIZE, as the minimum of either the size of the candidate server list itself, i.e., as specified in variable CANDIDATE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE, or the remaining number of servers requested by the client application after all the eligible servers have been selected, i.e., CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE - ELIGIBLE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE. Once block 725 has executed, then execution proceeds to decision block 730.

Detailed Description Text (38):
Blocks 730-775 collectively determine the weights for each of the eligible servers. Decision block 730 determines, based upon the number and type (candidate vis-a-vis eligible) of the presently available servers, whether a relatively large number of candidate, presumably black box servers, would be provided, i.e., servers for which no capacity information is currently known. If this is the case and particularly if

less than half of these collectively available systems have eligible servers, then to avoid taking undue risks in assigning work to all these servers, equal weights of one are set for all the servers that will be returned, from both server lists, to the client application. Alternatively, if more than half of these collective systems have eligible servers, i.e., for which capacity information is available, then, with relatively little risk, the weights assigned to the eligible servers can be extended, as discussed below, to the candidate servers that will be returned to the client application. Accordingly, if more than half of the collective systems have candidate servers, then decision block 730 directs execution via its YES path 734, to block 735 which sets a flag, FIXED.sub.-- WEIGHT, which signifies that fixed weights are to be assigned to the candidate servers, to YES. Alternatively, if more than half of the collective systems have eligible servers, then, decision block 730 routes execution, via its NO path 732, to block 740 which, when executed, sets the FIXED.sub.-- WEIGHT flag to NO; hence, variable weights are subsequently set. Once this flag has been duly set by either block 735 or 740, execution proceeds to block 745.

Detailed Description Text (64):
Block 1215, when executed, sets variable SYSTEM to designate the first system in REMAINING.sub.-- SYSTEMS.sub.-- LIST. Thereafter, block 1215 removes the system now designated by variable SYSTEM, i.e., the current system, from REMAINING.sub.-- SYSTEMS.sub.-- LIST. Once this occurs, execution proceeds to decision block 1220 which tests whether the current system is presently active. If this system is not active, then this system is no longer considered with execution merely looping back, via NO path 1224 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if the current system is active, then decision block 1220 routes execution, via its YES path 1222, to decision block 1225. This latter decision block determines, from the GRRI table, whether that system has a real instance that supports the generic resource requested by the client application, i.e., whether an eligible application server resides on the system for the client application. If no such server exists, then the current system is not considered further with execution looping back, via NO path 1228 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. In the event that the current system does possess an eligible application server (real instance for the requested generic resource), then decision block 1225 routes execution, via its YES path 1226, to decision block 1230. This latter decision block ascertains whether this present system is goal-oriented, i.e., running in a goal mode. As noted above, all servers for which capacity information is not available, including non-goal oriented servers, are not selected at all for session placement. In any event, if the current system is not running in goal mode, then decision block 1230 routes execution, via NO path 1233, to block 1234 which, in turn, sets the eligible system list to empty. Execution then exits. Alternatively, if the current system is running in goal mode, then decision block 1230 routes execution, via YES path 1232, to decision block 1235. This latter decision block ascertains whether the current system, i.e., as identified in SYSTEM, which supports the desired generic resource, has an application server (real instance) that is meeting its goals. If no such application server residing on the current system is meeting its corresponding goals, then decision block 1235 routes execution, via NO path 1238, to decision block 1240. This latter decision block checks to determine whether any of the systems within the set of ELIGIBLE.sub.-- SYSTEMS, as previously chosen by this loop thus far, contains a system that does not have an application server that is meeting its goals. If such a system does not exist, i.e., at least one system has already been chosen that is meeting its goals and the current system is an inferior choice to the system already chosen, then the present system is not considered any further. In this case, execution then loops back, via NO path 1244 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if such a system exists, i.e., the current as well as all the systems selected thus far are not meeting their goals, then decision block 1240 routes execution to block 1245 which, when executed, adds the identification of the

current system to the set of selected eligible systems as it presently exists, i.e., ELIGIBLE.sub.-- SYSTEMS.rarw.ELIGIBLE.sub.-- SYSTEMS+SYSTEM. Once the list of eligible systems is updated, then execution then loops back, via path 1246 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on.

Detailed Description Text (86):
Accordingly, in selecting competing servers for session placement, a more refined selection criteria can be used than that described above, with the order of preferential server selection being: (a) first, those servers that are fully meeting their goals; (b) second, those servers that are marginally missing their goals but having requisite capacity at a sufficiently high numeric (low business) importance level, such as level six, or greater, or just unused capacity; (c) third, those servers that are meeting their goals but only have sufficient capacity at a higher (numerically lower) business importance level than those servers marginally missing their goals; and (d) fourth and finally, those servers that are not meeting their goals at all, i.e., by more than a marginal amount. By favoring those servers that are marginally missing their goals (servers (b)) but have capacity at a sufficiently low business importance level to receive new sessions (and work) over those goal-meeting servers that have capacity only at higher business importance levels (servers (c)), this criteria assures that work of higher business importance currently active or pending at the latter servers will not be readily displaced. Those marginal servers that do not possess requisite minimum capacity at a sufficiently low business importance level are classified in the last group of servers. Servers in this last group (servers (d)) are only selected as a last resort since any additional sessions placed thereat (with ensuing work requests) are likely to excessively burden these servers, thus degrading their throughput to the ultimate detriment of meeting the overall business goals of the sysplex. Here, too, servers in each of the four classes (a-d) would be chosen in ascending order of session counts, i.e. those servers having the fewest sessions would be chosen first. The numeric server performance measure (s) that defines "marginal" operation is implementationally dependent and may well be empirically defined.

US005881238A

United States Patent [19]

Aman et al.

[11] **Patent Number: 5,881,238**

[45] **Date of Patent: *Mar. 9, 1999**

[54] **SYSTEM FOR ASSIGNMENT OF WORK REQUESTS BY IDENTIFYING SERVERS IN A MULTISYSTEM COMPLEX HAVING A MINIMUM PREDEFINED CAPACITY UTILIZATION AT LOWEST IMPORTANCE LEVEL**

[75] Inventors: **Jeffrey David Aman**, Wappingers Falls, N.Y.; **Curt Lee Cotner**, Gilroy, Calif.; **Donna Ngar Ting Dillenberger**, Yorktown Heights; **David Bruce Emmes**, Poughkeepsie, both of N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[*] Notice: The term of this patent shall not extend beyond the expiration date of Pat. No. 5,603,029.

[21] Appl. No.: **797,899**

[22] Filed: **Feb. 10, 1997**

Related U.S. Application Data

[63] Continuation of Ser. No. 476,157, Jun. 7, 1995, Pat. No. 5,603,029.

[51] **Int. Cl.**^{6} **G06F 13/00**

[52] **U.S. Cl.** **395/200.56**; 395/674

[58] **Field of Search** 364/229, 284.4, 364/281.7; 395/200.56, 674, 500, 680

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,702,006	10/1972	Page	395/675
4,542,458	9/1985	Kitajima et al.	395/674
4,727,487	2/1988	Masui et al.	395/675
4,748,558	5/1988	Hirosawa et al.	395/675
5,031,089	7/1991	Liu et al.	395/200.56
5,325,525	6/1994	Shan et al.	395/674
5,341,477	8/1994	Pitkin et al.	395/200.56
5,504,894	4/1996	Ferguson et al.	707/2

OTHER PUBLICATIONS

Service Management, IBM Technical Disclosure Bulletin, pp. 2330–2338, Dec. 1973.

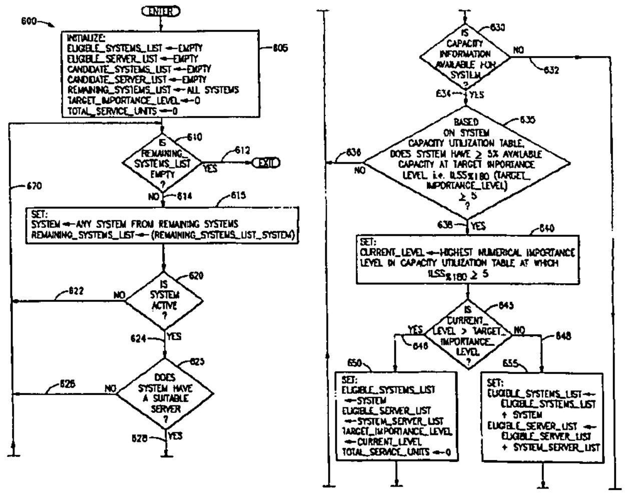
(List continued on next page.)

Primary Examiner—Thomas C. Lee
Assistant Examiner—Anderson I. Chen
Attorney, Agent, or Firm—W. A. Kinnaman, Jr.

[57] **ABSTRACT**

Apparatus and accompanying methods for use preferably in a multi-system shared data (sysplex) environment (**100**), wherein each system (**110**) provides one or more servers (**115**), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system (s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

25 Claims, 25 Drawing Sheets


ENTER
600
INITIALIZE:
ELIGIBLE_SYSTEMS_LIST ← EMPTY
ELIGIBLE_SERVER_LIST ← EMPTY
CANDIDATE_SYSTEMS_LIST ← EMPTY
CANDIDATE_SERVER_LIST ← EMPTY
REMAINING_SYSTEMS_LIST ← ALL SYSTEMS
TARGET_IMPORTANCE_LEVEL ← 0
TOTAL_SERVICE_UNITS ← 0
605
IS REMAINING_SYSTEMS_LIST EMPTY ?
610
YES
612
EXIT
670
NO
614
615
SET:
SYSTEM ← ANY SYSTEM FROM REMAINING SYSTEMS
REMAINING_SYSTEMS_LIST ← (REMAINING_SYSTEMS_LIST_SYSTEM)
IS SYSTEM ACTIVE ?
620
622
NO
624
YES
DOES SYSTEM HAVE A SUITABLE SERVER ?
625
626
NO
628
YES
IS CAPACITY INFORMATION AVAILABLE FOR SYSTEM ?
630
NO
632
634
YES
BASED ON SYSTEM CAPACITY UTILIZATION TABLE, DOES SYSTEM HAVE ≥ 5% AVAILABLE CAPACITY AT TARGET IMPORTANCE LEVEL, i.e. ILSS%180 (TARGET_IMPORTANCE_LEVEL) ≥ 5 ?
635
636
NO
638
YES
640
SET:
CURRENT_LEVEL ← HIGHEST NUMERICAL IMPORTANCE LEVEL IN CAPACITY UTILIZATION TABLE AT WHICH ILSS%180 ≥ 5
IS CURRENT_LEVEL > TARGET_IMPORTANCE_LEVEL ?
645
YES
646
NO
648
650
SET:
ELIGIBLE_SYSTEMS_LIST ← SYSTEM
ELIGIBLE_SERVER_LIST ← SYSTEM_SERVER_LIST
TARGET_IMPORTANCE_LEVEL ← CURRENT_LEVEL
TOTAL_SERVICE_UNITS ← 0
655
SET:
ELIGIBLE_SYSTEMS_LIST ← ELIGIBLE_SYSTEMS_LIST + SYSTEM
ELIGIBLE_SERVER_LIST ← ELIGIBLE_SERVER_LIST + SYSTEM_SERVER_LIST

First Hit   Fwd Refs

☐ Generate Collection Print

L19: Entry 10 of 13   File: USPT   Mar 9, 1999

DOCUMENT-IDENTIFIER: US 5881238 A
TITLE: System for assignment of work requests by identifying servers in a multisystem complex having a minimum predefined capacity utilization at lowest importance level

Abstract Text (1):
Apparatus and accompanying methods for use preferably in a multi-system shared data (sysplex) environment (100), wherein each system (110) provides one or more servers (115), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system(s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

Brief Summary Text (27):
However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement may be to rotate the server entries among the different systems. Owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Brief Summary Text (28):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight. The client may request a refresh of this balancing assessment on a periodic basis.

Detailed Description Text (23):
However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement here may be to rotate servers around different systems before the same system is selected again for a different server. In any event, owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Detailed Description Text (24):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight.

Detailed Description Text (34):
Once routine 600 completes processing, what remains is two lists of servers: a list of eligible servers, i.e., policy related servers for which capacity information is available, and a list of all candidate servers--the former list possessing available requisite capacity at the lowest business importance level, the latter list being servers, whether goal-oriented or not (i.e., running under a policy or being "black box") but for which no capacity information is available. Routine 700 is then executed, as noted above, to set a maximum limit on the number of servers that can be selected from either of the server lists and subsequently provided to the client application, and to determine the weights for the eligible servers.

Detailed Description Text (36):
Upon entry into routine 700, execution first proceeds to blocks 705-725 which collectively specify the maximum number of servers that can be selected from the candidate server list and subsequently provided to the client application. Specifically, execution first proceeds to block 705 which, when executed, sets the size of a list of output servers to be provided, by our inventive process, to the client application. This size (variable CANDIDATE.sub.-- COPY.sub.-- SIZE) is taken to be a minimum of the size, as specified by the client application itself in its work request, of its server list, i.e., variable CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE, or the size of the candidate server list itself, i.e., variable CANDIDATE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE. Thereafter, decision block 710 determines whether the eligible server list contains any servers. If not, then the only servers that can be selected are those in the candidate server list. Accordingly, if no eligible servers exist, i.e., ELIGIBLE.sub.-- SERVER.sub.-- LIST is empty, then execution exits, via YES path 712, from routine 700. Alternatively, if eligible servers do exist, then decision block 710 directs execution, via NO path 714, to decision block 715. This latter decision block, when executed, ascertains whether more servers exist in the eligible server list than are requested by the client, i.e., whether CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE.ltoreq.ELIGIBLE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE? If this is true, then only eligible servers and no candidate servers will be placed in the client server list. In this case, decision block 715 will route execution, via YES path 716, to block 720 which, when executed, will set the number of servers, specified in variable CANDIDATE.sub.-- COPY.sub.-- SIZE, to be selected from the candidate server list to zero. Alternatively, if an insufficient number of servers exists in the eligible server list to satisfy the requirements of the client application, thereby requiring some or all of the servers to be selected from the candidate

server list, decision block 715 will route execution, via NO path 718, to block 725. This latter block, when executed, will set the number of servers to be selected from the candidate server list, i.e., as specified in variable CANDIDATE.sub.-- COPY.sub.-- SIZE, as the minimum of either the size of the candidate server list itself, i.e., as specified in variable CANDIDATE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE, or the remaining number of servers requested by the client application after all the eligible servers have been selected, i.e., CLIENT.sub.-- SERVER.sub.-- LIST.sub.-- SIZE--ELIGIBLE.sub.-- SERVER.sub.-- LIST.sub.-- SIZE. Once block 725 has executed, then execution proceeds to decision block 730.

Detailed Description Text (37):
Blocks 730-775 collectively determine the weights for each of the eligible servers. Decision block 730 determines, based upon the number and type (candidate vis-a-vis eligible) of the presently available servers, whether a relatively large number of candidate, presumably black box servers, would be provided, i.e., servers for which no capacity information is currently known. If this is the case and particularly if less than half of these collectively available systems have eligible servers, then to avoid taking undue risks in assigning work to all these servers, equal weights of one are set for all the servers that will be returned, from both server lists, to the client application. Alternatively, if more than half of these collective systems have eligible servers, i.e., for which capacity information is available, then, with relatively little risk, the weights assigned to the eligible servers can be extended, as discussed below, to the candidate servers that will be returned to the client application. Accordingly, if more than half of the collective systems have candidate servers, then decision block 730 directs execution via its YES path 734, to block 735 which sets a flag, FIXED.sub.-- WEIGHT, which signifies that fixed weights are to be assigned to the candidate servers, to YES. Alternatively, if more than half of the collective systems have eligible servers, then, decision block 730 routes execution, via its NO path 732, to block 740 which, when executed, sets the FIXED.sub.-- WEIGHT flag to NO; hence, variable weights are subsequently set. Once this flag has been duly set by either block 735 or 740, execution proceeds to block 745.

Detailed Description Text (63):
Block 1215, when executed, sets variable SYSTEM to designate the first system in REMAINING.sub.-- SYSTEMS.sub.-- LIST. Thereafter, block 1215 removes the system now designated by variable SYSTEM, i.e., the current system, from REMAINING.sub.-- SYSTEMS.sub.-- LIST. Once this occurs, execution proceeds to decision block 1220 which tests whether the current system is presently active. If this system is not active, then this system is no longer considered with execution merely looping back, via NO path 1224 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if the current system is active, then decision block 1220 routes execution, via its YES path 1222, to decision block 1225. This latter decision block determines, from the GRRI table, whether that system has a real instance that supports the generic resource requested by the client application, i.e., whether an eligible application server resides on the system for the client application. If no such server exists, then the current system is not considered further with execution looping back, via NO path 1228 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. In the event that the current system does possess an eligible application server (real instance for the requested generic resource), then decision block 1225 routes execution, via its YES path 1226, to decision block 1230. This latter decision block ascertains whether this present system is goal-oriented, i.e., running in a goal mode. As noted above, all servers for which capacity information is not available, including non-goal oriented servers, are not selected at all for session placement. In any event, if the current system is not running in goal mode, then decision block 1230 routes execution, via NO path 1233, to block 1234 which, in turn, sets the eligible system list to empty. Execution then exits. Alternatively, if the current system is

running in goal mode, then decision block 1230 routes execution, via YES path 1232, to decision block 1235. This latter decision block ascertains whether the current system, i.e., as identified in SYSTEM, which supports the desired generic resource, has an application server (real instance) that is meeting its goals. If no such application server residing on the current system is meeting its corresponding goals, then decision block 1235 routes execution, via NO path 1238, to decision block 1240. This latter decision block checks to determine whether any of the systems within the set of ELIGIBLE.sub.-- SYSTEMS, as previously chosen by this loop thus far, contains a system that does not have an application server that is meeting its goals. If such a system does not exist, i.e., at least one system has already been chosen that is meeting its goals and the current system is an inferior choice to the system already chosen, then the present system is not considered any further. In this case, execution then loops back, via NO path 1244 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if such a system exists, i.e., the current as well as all the systems selected thus far are not meeting their goals, then decision block 1240 routes execution to block 1245 which, when executed, adds the identification of the current system to the set of selected eligible systems as it presently exists, i.e., ELIGIBLE.sub.-- SYSTEMS<-ELIGIBLE.sub.-- SYSTEMS+SYSTEM. Once the list of eligible systems is updated, then execution then loops back, via path 1246 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on.

Detailed Description Text (85):
Accordingly, in selecting competing servers for session placement, a more refined selection criteria can be used than that described above, with the order of preferential server selection being: (a) first, those servers that are fully meeting their goals; (b) second, those servers that are marginally missing their goals but having requisite capacity at a sufficiently high numeric (low business) importance level, such as level six, or greater, or just unused capacity; (c) third, those servers that are meeting their goals but only have sufficient capacity at a higher (numerically lower) business importance level than those servers marginally missing their goals; and (d) fourth and finally, those servers that are not meeting their goals at all, i.e., by more than a marginal amount. By favoring those servers that are marginally missing their goals (servers (b)) but have capacity at a sufficiently low business importance level to receive new sessions (and work) over those goal-meeting servers that have capacity only at higher business importance levels (servers (c)), this criteria assures that work of higher business importance currently active or pending at the latter servers will not be readily displaced. Those marginal servers that do not possess requisite minimum capacity at a sufficiently low business importance level are classified in the last group of servers. Servers in this last group (servers (d)) are only selected as a last resort since any additional sessions placed thereat (with ensuing work requests) are likely to excessively burden these servers, thus degrading their throughput to the ultimate detriment of meeting the overall business goals of the sysplex. Here, too, servers in each of the four classes (a-d) would be chosen in ascending order of session counts, i.e. those servers having the fewest sessions would be chosen first. The numeric server performance measure(s) that defines "marginal" operation is implementationally dependent and may well be empirically defined.

US006351775B1

(12) **United States Patent**
Yu

(10) **Patent No.: US 6,351,775 B1**
(45) **Date of Patent: Feb. 26, 2002**

(54) LOADING BALANCING ACROSS SERVERS IN A COMPUTER NETWORK

(75) Inventor: **Philip Shi-Lung Yu**, Chappaqua, NY (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/866,461**

(22) Filed: **May 30, 1997**

(51) **Int. Cl.**^{7} **G06F 15/173**

(52) **U.S. Cl.** **709/238**; 709/239; 709/240; 709/241; 709/242; 370/237; 370/400

(58) **Field of Search** 709/242, 239, 709/240, 241, 238; 370/237, 400

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,371,852 A		12/1994	Attanasio et al.	
5,517,620 A	*	5/1996	Hashimoto et al.	709/242
5,526,414 A	*	6/1996	Bedard et al.	379/221
5,544,313 A	*	8/1996	Shachnai et al.	709/219
5,828,847 A	*	10/1998	Gehr et al.	709/239
5,864,535 A	*	1/1999	Basilico	370/231
5,930,348 A	*	7/1999	Regnier et al.	379/221
6,078,943 A	*	6/2000	Yu	709/105
6,091,720 A	*	7/2000	Bedard et al.	370/351

FOREIGN PATENT DOCUMENTS

JP	8-214063	8/1996

OTHER PUBLICATIONS

M. Colajanni et al., "Scheduling Algorithms for Distributed Web Servers", RC 20680 (91683) Jan. 6, 1997, Computer Science/Mathematics, Research Report, 29 pages.

T. Brisco, "DNS Support for Load Balancing", Apr. 1995, 6 pages, Network Working Group, Rutgers University.

Daniel M. Dias et al., "A Scalable and Highly Available Web Server", (not dated), 8 pages, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, N. Y. 10598.

Eri D. Katz et al., "A scalable HTTP server: The NCSA prototype", 1994, pp. 155–164, vol. 27, Computer Networks and ISDN Systems.

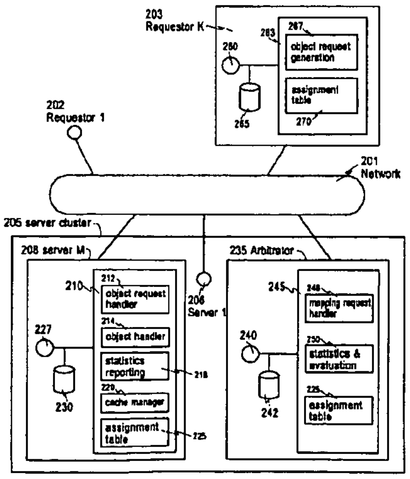
* cited by examiner

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—F. Chau & Associates, LLP

(57) **ABSTRACT**

A dynamic routing of object requests among a collection or cluster of servers factors the caching efficiency of the servers and the load balance or just the load balance. The routing information on server location can be dynamically updated by piggybacking meta information with the request response. To improve the cache hit at the server, the server selection factors the identifier (e.g. URL) of the object requested. A partitioning method can map object identifiers into classes; and requester nodes maintain a server assignment table to map each class into a server selection. The class-to-server assignment table can change dynamically as the workload varies and also factors the server capacity. The requester node need only be informed on an "on-demand" basis on the dynamic change of the class-to-server assignment (and thus reduce communication traffic). In the Internet, the collection of servers can be either a proxy or Web server cluster and can include a DNS and/or TCP-router. The PICS protocol can be used by the server to provide the meta information on the "new" class-to-server mapping when a request is directed to a server based on an invalid or obsolete class-to-server mapping. DNS based routing for load balancing of a server cluster can also benefit. By piggybacking meta data with the returned object to reassign the requester to another server for future requests, adverse effects of the TTL on the load balance are overcome without increasing traffic.

75 Claims, 15 Drawing Sheets



First Hit Fwd Refs

☐ Generate Collection Print

L19: Entry 4 of 13 File: USPT Feb 26, 2002

DOCUMENT-IDENTIFIER: US 6351775 B1
TITLE: Loading balancing across servers in a computer network

Abstract Text (1):
A dynamic routing of object requests among a collection or cluster of servers factors the caching efficiency of the servers and the load balance or just the load balance. The routing information on server location can be dynamically updated by piggybacking meta information with the request response. To improve the cache hit at the server, the server selection factors the identifier (e.g. URL) of the object requested. A partitioning method can map object identifiers into classes; and requester nodes maintain a server assignment table to map each class into a server selection. The class-to-server assignment table can change dynamically as the workload varies and also factors the server capacity. The requester node need only be informed on an "on-demand" basis on the dynamic change of the class-to-server assignment (and thus reduce communication traffic). In the Internet, the collection of servers can be either a proxy or Web server cluster and can include a DNS and/or TCP-router. The PICS protocol can be used by the server to provide the meta information on the "new" class-to-server mapping when a request is directed to a server based on an invalid or obsolete class-to-server mapping. DNS based routing for load balancing of a server cluster can also benefit. By piggybacking meta data with the returned object to reassign the requester to another server for future requests, adverse effects of the TTL on the load balance are overcome without increasing traffic.

Detailed Description Text (19):
FIG. 10 depicts an example of the Statistics and Evaluation routine (250). As depicted, in step 1010, statistics collection requests are communicated to all servers to get the CS(j,i), for i=1, . . . , N, from server j, for j=1, . . . , M. In step 1020, CA(i) is calculated for each class (the total number of requests across all servers for each class i). In step 1030, SA(j) is calculated for each server j, (the total number of requests to the assigned classes of each server j). In step 1040, the upper threshold, TH, of the load on a server is calculated. TH is preferably defined to be a fraction (d) above the average load. For example, d can be 0.2, which means, the target for load balancing is to have none of the servers exceeding 20% of the average. In step 1050, if any server's load exceeds the threshold TH, the Reassignment Routine is invoked to adjust the class-to-server assignment so that better load balancing can be achieved. A detailed example of the Reassignment Routine will be described with reference to FIG. 11. In step 1070, the statistics collection timer is reset to the length of the desired statistics collection interval.

Detailed Description Text (30):
In a preferred embodiment, the DNS (167) collects the number of requests issued from each requester and will generate a requester-to-server assignment table to balance the load among the servers. (For heterogeneous servers, the assigned load can be made proportional to the server's processing capacity). When a (name-to-address) mapping request arrives at the DNS (167), a server (161 . . . 163) is assigned based on the requester name (or IP address) in the assignment table. The mapping is hierarchical and multi-level, e.g., URL=>Class=>virtual server=>server. The DNS (167) can collect the load statistics and update the assignment table (225)

based on a measurement interval (much) smaller than the TTL. Thus, a new assignment table can be quickly generated, to better reflect load conditions. All servers (161 . . . 163) get the up-to-date version of the assignment table (225) from the DNS (167). As before, the requesters (110 . . . 153) need not be informed of the change; they can still send requests based on the previous (name-to-address) mapping. However, if a server receives a request from a requester that is no longer assigned to that server, the server will inform the requester of the server (161 . . . 163) to which future requests should be issued. The current request will still be served and the new assignment information can be piggybacked, e.g., using PICS or a similar mechanism, with the response or returned object. When a server is overloaded, it can send an alarm signal to the DNS (167). Each time an alarm is received, the DNS (167) can recalculate the assignment table to reduce the number of requesters assigned to any overloaded servers. The requesters can also be partitioned into classes so that the assignment table can then become a class-to-server assignment.

US006324580B1

(12) **United States Patent**
Jindal et al.

(10) **Patent No.: US 6,324,580 B1**
(45) **Date of Patent: *Nov. 27, 2001**

(54) LOAD BALANCING FOR REPLICATED SERVICES

(75) Inventors: **Anita Jindal**; **Swee Boon Lim**, both of Cupertino; **Sanjay Radia**, Fremont; **Whei-Ling Chang**, Saratoga, all of CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/146,848**

(22) Filed: **Sep. 3, 1998**

(51) **Int. Cl.**^{7} **G06F 15/16**; G06F 9/00
(52) **U.S. Cl.** **709/228**; 709/105
(58) **Field of Search** 709/223, 232, 709/238, 244, 105, 102, 228; 370/232; 712/27

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,371,852	12/1994	Attanasio et al.	395/200
5,459,837	10/1995	Caccavale	395/184.01
5,583,994	12/1996	Rangan	395/200
5,742,598	4/1998	Dunn et al.	370/393
5,774,660 *	6/1998	Brendel et al.	709/201

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0715257 A1	6/1996	(EP)	G06F/9/46
WO 98/26559	6/1998	(WO)	H04L/29/12

OTHER PUBLICATIONS

Colajanni, M., Yu, P. and Dias, D., Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems, IEEE Transactions on Parallel and Distributed Systems, vol. 9, No. 6, Jun. 1998, pp. 585–599.

Pending, U.S. Patent Application Serial No. 09/146,771, by Anita Jindal, et al., titled "System for Responding to a Resource Request," filed Sep. 3, 1998, with Attorney Docket No. SUN-P3317-JTF.

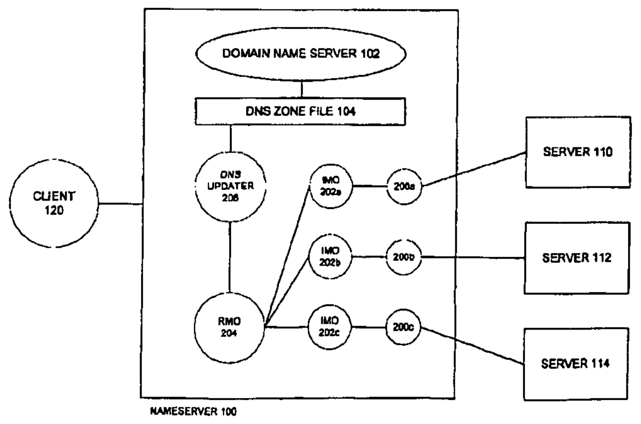
Pending U.S. Patent Application Serial No. 09/146,772, by Anita Jindal, et al., titled "Load Balancing in a Network Environment," filed Sep. 3, 1998, with Attorney Docket No. SUN-P3374-JTF.

Primary Examiner—Zarni Maung
Assistant Examiner—Jason D. Cardone
(74) *Attorney, Agent, or Firm*—Park, Vaughan & Fleming LLP

(57) **ABSTRACT**

A method is provided for load balancing requests for a replicated service or application among a plurality of servers operating instances of the replicated service or application. A policy is selected for choosing a preferred server from the plurality of servers according to one or more specified status or operational characteristics of the servers, such as the least-loaded or closest server. The policy is encapsulated within multiple levels of objects or modules that are distributed among the servers offering the replicated service and a central server that receives requests for the service. Status objects gather or retrieve information concerning the specified status or operational characteristic(s) of each of the plurality of servers. An individual server monitor object operates for each instance of the replicated service to invoke one or more status objects and receive the necessary information. A central replicated monitor object receives the information from each individual server monitor object. The information from the servers is analyzed to select the server having the optimal status or operational characteristic(s). An update object updates the central server, such as a domain name server, to indicate the preferred server. Requests for the replicated service are then directed to the preferred server until a different preferred server is identified.

30 Claims, 5 Drawing Sheets



First Hit Fwd Refs

☐ Generate Collection Print

L19: Entry 5 of 13 File: USPT Nov 27, 2001

DOCUMENT-IDENTIFIER: US 6324580 B1
TITLE: Load balancing for replicated services

Abstract Text (1):
A method is provided for load balancing requests for a replicated service or application among a plurality of servers operating instances of the replicated service or application. A policy is selected for choosing a preferred server from the plurality of servers according to one or more specified status or operational characteristics of the servers, such as the least-loaded or closest server. The policy is encapsulated within multiple levels of objects or modules that are distributed among the servers offering the replicated service and a central server that receives requests for the service. Status objects gather or retrieve information concerning the specified status or operational characteristic(s) of each of the plurality of servers. An individual server monitor object operates for each instance of the replicated service to invoke one or more status objects and receive the necessary information. A central replicated monitor object receives the information from each individual server monitor object. The information from the servers is analyzed to select the server having the optimal status or operational characteristic(s). An update object updates the central server, such as a domain name server, to indicate the preferred server. Requests for the replicated service are then directed to the preferred server until a different preferred server is identified.

Detailed Description Text (44):
In contrast, if an intrusive mode of operation is to be used, a status object reflecting a fastest-response policy is constructed to execute on a server (e.g., a server offering the replicated service) in order to ascertain the number of requests pending on the server.

Detailed Description Text (46):
In state 508 individual monitor objects are created or invoked. Illustratively, one IMO is generated for each instance of a replicated service. Depending upon whether the replicated service is to be load-balanced intrusively or non-intrusively, the IMO objects are installed on either the individual servers offering the service, the DNS server, or some intermediate computer system. As described above, IMO objects may be configured to invoke one or more status objects and collect and report certain information or data. The collected information may include a server's load (e.g., number of requests waiting and/or being processed), capacity (e.g., the number of requests the server can handle), operational status (e.g., whether the server is up or down), etc.

US006249800B1

(12) **United States Patent**
Aman et al.

(10) **Patent No.: US 6,249,800 B1**
(45) **Date of Patent: Jun. 19, 2001**

(54) **APPARATUS AND ACCOMPANYING METHOD FOR ASSIGNING SESSION REQUESTS IN A MULTI-SERVER SYSPLEX ENVIRONMENT**

(75) Inventors: **Jeffrey David Aman**, Wappingers Falls; **David Bruce Emmes**, Poughkeepsie; **David Walsh Palmieri**, Wappingers Falls, all of NY (US)

(73) Assignee: **International Business Machines Corporartion**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/488,374**

(22) Filed: **Jun. 7, 1995**

(51) **Int. Cl.**^{7} **G06F 15/16**
(52) **U.S. Cl.** **709/105**; 709/239; 709/229
(58) **Field of Search** 395/600, 700, 395/650, 671, 672, 673; 709/224, 239, 232, 102, 103, 104, 223, 105, 203

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,858,120 *	8/1989	Samuelson	364/401
5,155,851 *	10/1992	Krishnan	395/650
5,341,477 *	8/1994	Pitkin et al.	395/200
5,475,819 *	12/1995	Miller et al.	395/200.03
5,504,894 *	4/1996	Ferguson et al.	395/650
5,548,724 *	8/1996	Akizawa et al.	709/203
5,572,528 *	11/1996	Shuen	370/85.13
5,603,029 *	2/1997	Aman et al.	709/105
5,781,743 *	7/1998	Matsuno et al.	709/228
5,802,301 *	9/1998	Dan et al.	709/223
5,881,238 *	3/1999	Aman et al.	709/226

OTHER PUBLICATIONS

"A CAD Package for simulation of Parellel Processing architectures" H. Diab et al, Computer–Aided Engineering Journal, Aug. 1990.

"A Scalable Sharing Architecture for a Parallel Database System" Vibby Gottenukkala et al., Parallel and Distributed Processing, 1994.

"A Task Migration algorithm for load balancing in a distributed system" by Jea–Cheal Ryou, Johnny S. K. Wong, System Sciences, 1989 Annul, Hawaii Intr'l. .*

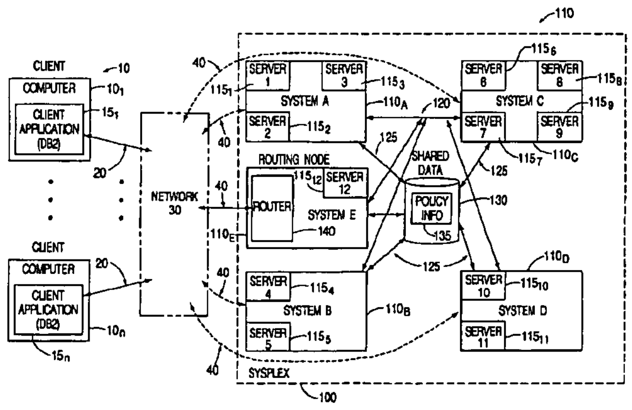
(List continued on next page.)

Primary Examiner—Larry D. Donaghue
(74) *Attorney, Agent, or Firm*—Michaelson & Wallace; Peter L. Michaelson; William A. Kinnaman, Jr.

(57) **ABSTRACT**

Apparatus and accompanying methods for use preferably in a multi-system shared data (sysplex) environment (**100**), wherein each system (**110**) provides one or more servers (**115**), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system (s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

20 Claims, 25 Drawing Sheets



First Hit Fwd Refs

☐ Generate Collection Print

L19: Entry 6 of 13 File: USPT Jun 19, 2001

DOCUMENT-IDENTIFIER: US 6249800 B1
TITLE: Apparatus and accompanying method for assigning session requests in a multi-server sysplex environment

Abstract Text (1):
Apparatus and accompanying methods for use preferably in a multi-system shared data (sysplex) environment (100), wherein each system (110) provides one or more servers (115), for dynamically and adaptively assigning and balancing new work and for new session requests, among the servers in the sysplex, in view of attendant user-defined business importance of these requests and available sysplex resource capacity so as to meet overall business goals. Specifically, systems and servers are categorized into two classes: eligible, i.e., goal-oriented servers running under a policy and for which capacity information is currently available, and candidate, i.e., servers which lack capacity information. Work requests for a client application are assigned first to various eligible systems and eligible servers thereon based on their current capacity to accept new work and in a manner that meets business goals inherent in a sysplex policy; followed, if additional servers are requested by that application, to candidate systems and candidate servers thereon. As to session placement, first those system(s) are selected that have lowest utilization, at a target importance level, but with sufficient available capacity at that level. Competing servers on the selected system(s) are then evaluated based on their corresponding session count data to yield a single resulting server. Thereafter, identification of multiple servers and their corresponding weights are returned to, e.g., a client application, for eventual routing of work requests to those servers, or the identification of a single server is returned to that application for establishing a new session therewith.

Brief Summary Text (28):
However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement may be to rotate the server entries among the different systems. Owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Brief Summary Text (29):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight. The client may request a refresh of this balancing assessment on a periodic basis.

Detailed Description Text (25):

However, if the eligible servers are only those with zero weights--i.e., those eligible servers with relatively little capacity, then each of these eligible servers are successively assigned a common weight of one and selected in seriatim until the identifications of all these servers (including their weights) have been written into the client server list. An improvement here may be to rotate servers around different systems before the same system is selected again for a different server. In any event, owing to their apparent inability to handle more than a small number, if any, of additional work requests, hence none of these servers is now particularly favored for new work. Thereafter, if any candidate servers are to be selected, each of these servers are successively assigned a weight of one and selected, again in seriatim, until all the candidate servers are selected or the client server list becomes full, whichever occurs first. In view of a lack of capacity information, none of these candidate servers is particularly favored as well.

Detailed Description Text (26):
As a result of our inventive work request assignment method, a list of servers and their corresponding server weights is identified to the client application which, in turn, will directly route a portion of the total work requests thereat to each of these servers in proportion to its weight.

Detailed Description Text (39):
Once routine 600 completes processing, what remains is two lists of servers: a list of eligible servers, i.e., policy related servers for which capacity information is available, and a list of all candidate servers--the former list possessing available requisite capacity at the lowest business importance level, the latter list being servers, whether goal-oriented or not (i.e., running under a policy or being "black box") but for which no capacity information is available. Routine 700 is then executed, as noted above, to set a maximum limit on the number of servers that can be selected from either of the server lists and subsequently provided to the client application, and to determine the weights for the eligible servers.

Detailed Description Text (41):
Upon entry into routine 700, execution first proceeds to blocks 705-725 which collectively specify the maximum number of servers that can be selected from the candidate server list and subsequently provided to the client application. Specifically, execution first proceeds to block 705 which, when executed, sets the size of a list of output servers to be provided, by our inventive process, to the client application. This size (variable CANDIDATE_COPY_SIZE) is taken to be a minimum of the size, as specified by the client application itself in its work request, of its server list, i.e., variable CLIENT_SERVER_LIST_SIZE, or the size of the candidate server list itself, i.e., variable CANDIDATE_SERVER_LIST_SIZE. Thereafter, decision block 710 determines whether the eligible server list contains any servers. If not, then the only servers that can be selected are those in the candidate server list. Accordingly, if no eligible servers exist, i.e., ELIGIBLE_SERVER_LIST is empty, then execution exits, via YES path 712, from routine 700. Alternatively, if eligible servers do exist, then decision block 710 directs execution, via NO path 714, to decision block 715. This latter decision block, when executed, ascertains whether more servers exist in the eligible server list than are requested by the client, i.e., whether CLIENT_SERVER_LIST_SIZE.ltoreq.ELIGIBLE_SERVER_LIST_SIZE? If this is true, then only eligible servers and no candidate servers will be placed in the client server list. In this case, decision block 715 will route execution, via YES path 716, to block 720 which, when executed, will set the number of servers, specified in variable CANDIDATE_COPY_SIZE, to be selected from the candidate server list to zero. Alternatively, if an insufficient number of servers exists in the eligible server list to satisfy the requirements of the client application, thereby requiring some or all of the servers to be selected from the candidate server list, decision block 715 will route execution, via NO path 718, to block 725. This latter block, when executed, will set the number of servers to be selected from the

candidate server list, i.e., as specified in variable CANDIDATE_COPY_SIZE, as the minimum of either the size of the candidate server list itself, i.e., as specified in variable CANDIDATE_SERVER_LIST_SIZE, or the remaining number of servers requested by the client application after all the eligible servers have been selected, i.e., CLIENT_SERVER_LIST_SIZE--ELIGIBLE_SERVER_LIST_SIZE. Once block 725 has executed, then execution proceeds to decision block 730.

Detailed Description Text (42):
Blocks 730-775 collectively determine the weights for each of the eligible servers. Decision block 730 determines, based upon the number and type (candidate vis-a-vis eligible) of the presently available servers, whether a relatively large number of candidate, presumably black box servers, would be provided, i.e., servers for which no capacity information is currently known. If this is the case and particularly if less than half of these collectively available systems have eligible servers, then to avoid taking undue risks in assigning work to all these servers, equal weights of one are set for all the servers that will be returned, from both server lists, to the client application. Alternatively, if more than half of these collective systems have eligible servers, i.e., for which capacity information is available, then, with relatively little risk, the weights assigned to the eligible servers can be extended, as discussed below, to the candidate servers that will be returned to the client application. Accordingly, if more than half of the collective systems have candidate servers, then decision block 730 directs execution via its YES path 734, to block 735 which sets a flag, FIXED_WEIGHT, which signifies that fixed weights are to be assigned to the candidate servers, to YES. Alternatively, if more than half of the collective systems have eligible servers, then, decision block 730 routes execution, via its NO path 732, to block 740 which, when executed, sets the FIXED_WEIGHT flag to NO; hence, variable weights are subsequently set. Once this flag has been duly set by either block 735 or 740, execution proceeds to block 745.

Detailed Description Text (72):
Block 1215, when executed, sets variable SYSTEM to designate the first system in REMAINING_SYSTEMS_LIST. Thereafter, block 1215 removes the system now designated by variable SYSTEM, i.e., the current system, from REMAINING_SYSTEMS_LIST. Once this occurs, execution proceeds to decision block 1220 which tests whether the current system is presently active. If this system is not active, then this system is no longer considered with execution merely looping back, via NO path 1224 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if the current system is active, then decision block 1220 routes execution, via its YES path 1222, to decision block 1225. This latter decision block determines, from the GRRI table, whether that system has a real instance that supports the generic resource requested by the client application, i.e., whether an eligible application server resides on the system for the client application. If no such server exists, then the current system is not considered further with execution looping back, via NO path 1228 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. In the event that the current system does possess an eligible application server (real instance for the requested generic resource), then decision block 1225 routes execution, via its YES path 1226, to decision block 1230. This latter decision block ascertains whether this present system is goal-oriented, i.e., running in a goal mode. As noted above, all servers for which capacity information is not available, including non-goal oriented servers, are not selected at all for session placement. In any event, if the current system is not running in goal mode, then decision block 1230 routes execution, via NO path 1233, to block 1234 which, in turn, sets the eligible system list to empty. Execution then exits. Alternatively, if the current system is running in goal mode, then decision block 1230 routes execution, via YES path 1232, to decision block 1235. This latter decision block ascertains whether the current system, i.e., as identified in SYSTEM, which supports the desired generic resource, has an application server (real instance) that is meeting its goals. If no such

application server residing on the current system is meeting its corresponding goals, then decision block 1235 routes execution, via NO path 1238, to decision block 1240. This latter decision block checks to determine whether any of the systems within the set of ELIGIBLE_SYSTEMS, as previously chosen by this loop thus far, contains a system that does not have an application server that is meeting its goals. If such a system does not exist, i.e., at least one system has already been chosen that is meeting its goals and the current system is an inferior choice to the system already chosen, then the present system is not considered any further. In this case, execution then loops back, via NO path 1244 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on. Alternatively, if such a system exists, i.e., the current as well as all the systems selected thus far are not meeting their goals, then decision block 1240 routes execution to block 1245 which, when executed, adds the identification of the current system to the set of selected eligible systems as it presently exists, i.e., ELIGIBLE_SYSTEMS<-ELIGIBLE_SYSTEMS+SYSTEM. Once the list of eligible systems is updated, then execution then loops back, via path 1246 and path 1262, to decision block 1210 to check whether any further systems remain to be considered and so on.

Detailed Description Text (95):
Accordingly, in selecting competing servers for session placement, a more refined selection criteria can be used than that described above, with the order of preferential server selection being: (a) first, those servers that are fully meeting their goals; (b) second, those servers that are marginally missing their goals but having requisite capacity at a sufficiently high numeric (low business) importance level, such as level six, or greater, or just unused capacity; (c) third, those servers that are meeting their goals but only have sufficient capacity at a higher (numerically lower) business importance level than those servers marginally missing their goals; and (d) fourth and finally, those servers that are not meeting their goals at all, i.e., by more than a marginal amount. By favoring those servers that are marginally missing their goals (servers (b)) but have capacity at a sufficiently low business importance level to receive new sessions (and work) over those goal-meeting servers that have capacity only at higher business importance levels (servers (c)), this criteria assures that work of higher business importance currently active or pending at the latter servers will not be readily displaced. Those marginal servers that do not possess requisite minimum capacity at a sufficiently low business importance level are classified in the last group of servers. Servers in this last group (servers (d)) are only selected as a last resort since any additional sessions placed thereat (with ensuing work requests) are likely to excessively burden these servers, thus degrading their throughput to the ultimate detriment of meeting the overall business goals of the sysplex. Here, too, servers in each of the four classes (a-d) would be chosen in ascending order of session counts, i.e. those servers having the fewest sessions would be chosen first. The numeric server performance measure(s) that defines "marginal" operation is implementationally dependent and may well be empirically defined.

US006047309A

United States Patent [19]

Dan et al.

[11] **Patent Number: 6,047,309**

[45] **Date of Patent: Apr. 4, 2000**

[54] **RECORDING OBSERVED AND REPORTED RESPONSE CHARACTERISTICS AT SERVER AND/OR CLIENT NODES IN A REPLICATED DATA ENVIRONMENT, AND SELECTING A SERVER TO PROVIDE DATA BASED ON THE OBSERVED AND/OR REPORTED RESPONSE CHARACTERISTICS**

[75] Inventors: **Asit Dan**, West Harrison; **Martin G. Kienzle**, Somers; **Dinkar Sitaram**, Yorktown Heights; **William H. Tetzlaff**, Mount Kisco, all of N.Y.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/537,025**

[22] Filed: **Oct. 2, 1995**

[51] **Int. Cl.**7 .... **G06F 13/14**

[52] **U.S. Cl.** **709/201**; 709/243; 709/105; 709/203; 348/7

[58] **Field of Search** 348/7; 395/200.01, 395/200.11, 835, 837, 838, 675; 709/201, 202, 203, 224–226, 223, 241, 105, 243

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,031,089	7/1991	Liu et al.	709/226
5,283,897	2/1994	Georgiadis et al.	395/675
5,341,477	8/1994	Pitkin et al.	395/200.56
5,371,532	12/1994	Gelman et al.	348/7
5,414,455	5/1995	Hopper et al.	348/7
5,459,837	10/1995	Caccavale	395/184.01
5,530,557	6/1996	Asit et al.	386/125
5,544,313	8/1996	Shachnai et al.	395/200.01
5,544,327	8/1996	Dan et al.	395/200.64
5,548,724	8/1996	Akizawa et al.	395/200.03
5,553,224	9/1996	Russell et al.	395/200.57
5,561,637	10/1996	Dan et al.	365/230.03
5,577,251	11/1996	Hamilton et al.	395/671
5,583,994	12/1996	Rangan	395/200.09
5,590,117	12/1996	Iida et al.	370/248
5,610,841	3/1997	Tanaka et al.	395/200.49
5,633,924	5/1997	Kaish et al.	379/266
5,644,714	7/1997	Kikinis	395/200.49
5,664,106	9/1997	Caccavale	395/200.54
5,687,372	11/1997	Hotea et al.	395/675
5,774,668	6/1998	Choquier et al.	395/200.53
5,850,517	12/1998	Verkler et al.	395/200.32

OTHER PUBLICATIONS

P.B. Berra et al., "Issues in Networking and Data Management of Distributed Multimedia Systems", 1992 IEEE, pp 4–15.

S.L. hardt–Kornacki et al., "Optimization Model for the Delivery of Interactive Multimedia Documents", 1991, Globecom '91, pp 669–673 (20.2.1–20.25).

Thomas D.C. Little, "Network Considerations for Distributed Multimedia Object Composition and Communication", Nov. 1990, IEEE Network Magazine, pp 32–49.

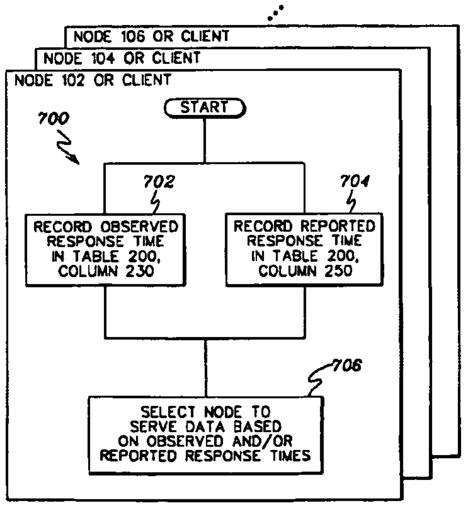
Domenico Ferrari, "Client Requirements for Real–Time Communication Services", Nov. 1990, IEEE Communications Magazine, pp 65–72.

Primary Examiner—Eddie P. Chan
Assistant Examiner—Hong Kim
Attorney, Agent, or Firm—Heslin & Rothenberg, P.C.

[57] **ABSTRACT**

A system and method for use in a distributed video-on-demand system of a type wherein at least one node provides blocks of video data to clients and wherein at least some of the blocks of video data are replicated on multiple nodes. Observed response characteristics for other nodes are recorded at least a given one of the nodes which serves a client requesting a replicated block. The given one of the nodes also records response characteristics reported to it by the other nodes. The node from which to fetch the replicated data block is selected based on which nodes include a copy of the replicated data block and based on at least one of the observed response characteristics and the reported response characteristics.

33 Claims, 8 Drawing Sheets


NODE 106 OR CLIENT
NODE 104 OR CLIENT
NODE 102 OR CLIENT
700
START
702
704
RECORD OBSERVED RESPONSE TIME IN TABLE 200, COLUMN 230
RECORD REPORTED RESPONSE TIME IN TABLE 200, COLUMN 250
706
SELECT NODE TO SERVE DATA BASED ON OBSERVED AND/OR REPORTED RESPONSE TIMES

First Hit Fwd Refs

☐ Generate Collection Print

L19: Entry 8 of 13 File: USPT Apr 4, 2000

DOCUMENT-IDENTIFIER: US 6047309 A
TITLE: Recording observed and reported response characteristics at server and/or client nodes in a replicated data environment, and selecting a server to provide data based on the observed and/or reported response characteristics

Detailed Description Text (11):
Each entry in the Server Load Table 200 tracks (stores) the observed response time and load of each block server 138 in the VOD System Node. Each row contains the serverId 210, the observed delay 230 of the last response from that server, the delay timestamp 220 containing the time at which the delay 230 was observed, and the server load 250 as reported by the server (e.g. server utilization). The load timestamp 240 is the server generated time stamp of the time at which the load 250 was reported. The load can be, for example, defined as the queue length (number of outstanding requests) at the server or as the server utilization measured as the number of requests served per unit time.